

# Real-Time Rate-Distortion Optimized Streaming of Wireless Video

Ahmed Abdelhadi, Andreas Gerstlauer and Sriram Vishwanath

## Abstract

Mobile cyberphysical systems have received considerable attention over the last decade, as communication, computing and control come together on a common platform. Understanding the complex interactions that govern the behavior of large complex cyberphysical systems is not an easy task. The goal of this paper is to address this challenge in the particular context of multimedia delivery over an autonomous aerial vehicle (AAV) network. Bandwidth requirements and stringent delay constraints of real-time video streaming, paired with limitations on computational complexity and power consumptions imposed by the underlying implementation platform, make cross-layer and cross-domain co-design approaches a necessity. In this paper, we propose a novel, low-complexity rate-distortion optimized (RDO) algorithms specifically targeted at video streaming over mobile embedded networks. We test the performance of our RDO algorithms using a network of AAVs both in simulation and implementation.

## I. INTRODUCTION

### A. Motivation

An explosion of interest in multimedia systems in the last decade has resulted in the need for developing efficient protocols for the delivery of rich content (such as video) across a wireless mobile network. Such media is delay sensitive while being both computationally and bandwidth intensive, and it poses considerable challenges in guaranteeing its reliable delivery. The main features that make real-time packetized media delivery particularly challenging are [24]:

- 1) *High data rate*: media (especially video) requires high data rate even when the physical, wireless link between nodes is rapidly changing over time.
- 2) *Real-time constraints*: there is a time-to-live (TTL) associated with each packet, so a packet received after its TTL expires is lost. In addition, in live streaming, the video packets has to be delivered within a limited transmitter/camera to receiver/display latency. Otherwise, they will be dropped/lost.
- 3) *Dependencies between frames*: MPEG4-H.264, MPEG2 and other compressed video formats are characterized by inter-frame dependency. This has two effects: first, if some frames are not successfully received this will lead to dropping of other successfully received frames because of their interdependency. Second, the number of interdependent frames determines the extent of compression in the video.

Rich multimedia delivery is increasingly an integral component specifically in many embedded and cyberphysical system applications, which creates additional implementation challenges. Such systems often have to operate in

tightly constrained environments that severely limit the available computational performance or the amount of power that can be consumed. Given a vast array of possible embedded and cyberphysical system implementation options and parameters, analyzing and designing protocols and application algorithms<sup>1</sup> for them is a considerably daunting task that invariably requires integrated, cross-layer and cross-domain co-design approaches.

In this chapter, we target the analysis and co-design of multimedia delivery over a particular cyberphysical system consisting of a network of autonomous aerial vehicles (AAVs). Such systems are of considerable interest across multiple civilian and military applications, including search and rescue, perimeter monitoring and object tracking. A network of AAVs poses a vast array of challenges - mobility, tracking and collision avoidance are essential for the physical operation of each AAV, while coordination and teaming critical for the network to carry out the task at hand. Central to all of these challenges is the ability to exchange high bandwidth delay sensitive data between the nodes in the network as reliably and efficiently as possible.

Our ultimate goal is to develop algorithms that exploit the structure of multimedia to deliver them efficiently and reliably over an AAV network, and test them in a real-world setting using a testbed. We have developed our own low-complexity rate-distortion optimized (RDO) streaming algorithms, and show that they outperform other mechanisms in the context of Horus, a custom built AAV testbed [8]. In a first step, we have developed software simulations for the mobility and channel models between AAVs, and we tested both existing and our proposed RDO video streaming techniques using these simulation models. Results show that optimized streaming can result in much more reliable and efficient video delivery than traditional protocols, in variants both with or without feedback. In the second step, we have implemented the system in realistic settings and tested our proposed RDO protocol and other protocols for different video compression. We used both temporal and spatial distortion measures to select the most reliable and efficient protocol.

## *B. Related Work*

There is considerable existing literature on developing protocols for efficient data delivery over wireless networks. A majority of this literature tends to focus on sensor networks designed for static settings, where nodes sense physical quantities that undergo a gradual change over time, e.g. temperature. For these applications, only a low data rate is required. Increasingly, sensor networks research incorporates dynamic network topologies as well. In [28], [29], [34], the authors conduct an experimental analysis on a dynamic sensor network where nodes move in a large area gathering data and then sending the collected data when near an access point. In our network, the nodes move in a prespecified pattern and gather media, e.g. video signals, and communicate them through the wireless network to an access point in another network. Note that our network is dynamically changing rapidly and intended to support a much higher data rate than conventional sensor networks.

<sup>1</sup>The word "Algorithm" comes from the name Al-Khwarizmi (c. 780-850), a Muslim mathematician, astronomer, geographer and a scholar in the House of Wisdom in Baghdad. He wrote an algorithm for distributing the inheritance of the deceased to his relatives according to the rules of *Quran*.

Simultaneously, there is a growing body of work on media compression and streaming, both over wired and wireless networks. One of these research efforts is presented in [25], where the authors address the problem of streaming packetized media over a lossy network in a rate-distortion optimized way. In [25], simulation results show that systems based on rate-distortion optimization (RDO) algorithms have steady-state gains of more than 2-6 dB compared to systems that are not rate-distortion optimized. In this work, a simplified simulation model approximating real-world conditions is assumed, which is only a first step in measuring the performance and expected improvement an algorithm has over existing implementations. For wide-spread system deployment and evaluation of achievable gains of any algorithm, experiments and validations must be carried out in a realistic setting. Towards this goal, we model and deploy RDO implementations in the context of an actual AAV testbed, where realistic simulations are a first step followed by running physical experiments in the field. Furthermore, the original RDO algorithm presented in [25] is based on ideal assumptions, e.g. in terms of its implementability. We instead propose modified RDO versions that can be efficiently realized with little to no overhead as part of standard network stacks on restricted embedded platforms.

### C. Our Contributions

Our contributions in this chapter are summarized as:

- A new low-complexity RDO algorithm [19] using default Media Access Control Layer (MAC-L) ACKs, which is validated through real-world simulations and shown to outperform ACKed and not ACKed transmission.
- A RDO algorithm using MAC-L beacons in order to achieve optimized video streaming under even lower complexity, as validated through experiments on a real-world AAV testbed and shown to minimize drops.
- A co-design of a RDO algorithm with adaptive video encoding to achieve optimized video streaming, developed for both MPEG2 and MJPEG streaming and validated through experiments on a real-world AAV testbed, where it is shown to improve received video quality.

## II. RATE-DISTORTION OPTIMIZATION PROBLEM

The RDO problem aims to optimize the amount of distortion in a network against the rate. Distortion is defined as the degradation in media quality as packets are dropped. The rate represents the amount of data, i.e. the number of packets transmitted per unit time. The data packets that comprise a stream vary in their importance in contributing to output quality and, conversely, distortion. As such, RDO is concerned with deciding which packets to drop based on media quality metrics, measuring both the deviation from the source material and the bit cost for each possible decision outcome. In other words, the problem aims to solve the question of, *which* packets to select for transmission, *when* to transmit them, and *how* to transmit them (e.g., how many times), such that the expected distortion is minimized, subject to constraints on the expected rate.

In [25], the authors presented an algorithm that minimize the distortion  $D$  for a given rate  $R$ . This is done by minimizing the Lagrangian  $D + \lambda R$  for some Lagrange multiplier  $\lambda$ . This algorithm is based on off-line transmission policy computation and on-line transmission policy truncation. This problem is defined and solved for every data

unit (i.e. video packet)  $l$ . As such, there exist a Lagrange multiplier  $\lambda_l$  for every data unit  $l$ . Depending on the expected channel rate, the value  $\lambda_l$  is a packet threshold used to decide if data unit  $l$  is the optimal video packet for transmission at this time instant or not. Solving the rate distortion optimization problem is not efficient for embedded applications in a real-time setting as the transmission policy computation for every packet threshold  $\lambda_l$  is time consuming and needs to be performed off-line. Instead, we propose novel RDO algorithms with low computational complexity in which the transmission policy is computed online in real time. In addition, solving the rate-distortion optimization problem presented in [25] requires a mathematical channel model with parameters that are updated regularly. If used in a real-time system, this will add significant computational complexity. Instead of a mathematical channel model, we measure the channel state by using measurable physical quantities that are already available in default system operation and therefore will not require extra computation. We use two types of channel state feedback for our two proposed algorithms:

- 1) MAC-L ACKs<sup>2</sup>: In standard 802.11 wireless networks, the destination sends ACKs to the source when packets are successfully received. These ACKs are used in our algorithm at the source to measure the channel state. We call the low complexity RDO algorithm that uses MAC-L ACKs **LCRDO-Ack**.
- 2) MAC-L beacons<sup>3</sup>: In standard 802.11 wireless networks, stations send beacons. To further reduce complexity, these beacons can be used instead of more frequent ACKs to measure the channel state at the source. We call the low complexity RDO algorithm that uses MAC-L beacons **LCRDO-Beacon**.

In addition to basic LCRDO variants, we co-design LCRDO-Beacon with adaptive video encoding algorithms using both MPEG2 and MJPEG compressions. We call this third low complexity RDO algorithm with adaptive co-design **LCRDO-Adaptive**.

The proposed LCRDO-Ack, LCRDO-Beacon, LCRDO-Adaptive algorithms have two important characteristics:

- 1) Real-time compatibility, where the transmission policy/channel state is computed on-line.
- 2) Lower complexity in terms of computational processing requirements.

### III. TESTBED

We demonstrate real-world performance of our optimized RDO algorithms in the context of Horus, a testbed composed of a network of AAVs communicating wirelessly in an ad-hoc fashion. In our experimental analysis, the AAV nodes are equipped with video cameras and are capable of streaming packetized media between them. Our network consists of a fixed number of nodes, that are placed in a prespecified topology. In this network, sources are streaming video data in real time to destinations. The routing path is given a priori and the topology

<sup>2</sup>In 802.11 networks, a transmitting station can not listen for collisions while sending data, mainly because a station can not have its receiver on while transmitting a frame. As a result, the receiving station needs to send an acknowledgement (ACK) if it detects no errors in the received frame.

<sup>3</sup>In 802.11 networks, access points periodically broadcast a beacon. The radio network interface card (NIC) receives these beacons while scanning and takes note of the corresponding signal strengths. The beacons contain information about the access point, including service set identifier (SSID), supported data rates, etc.

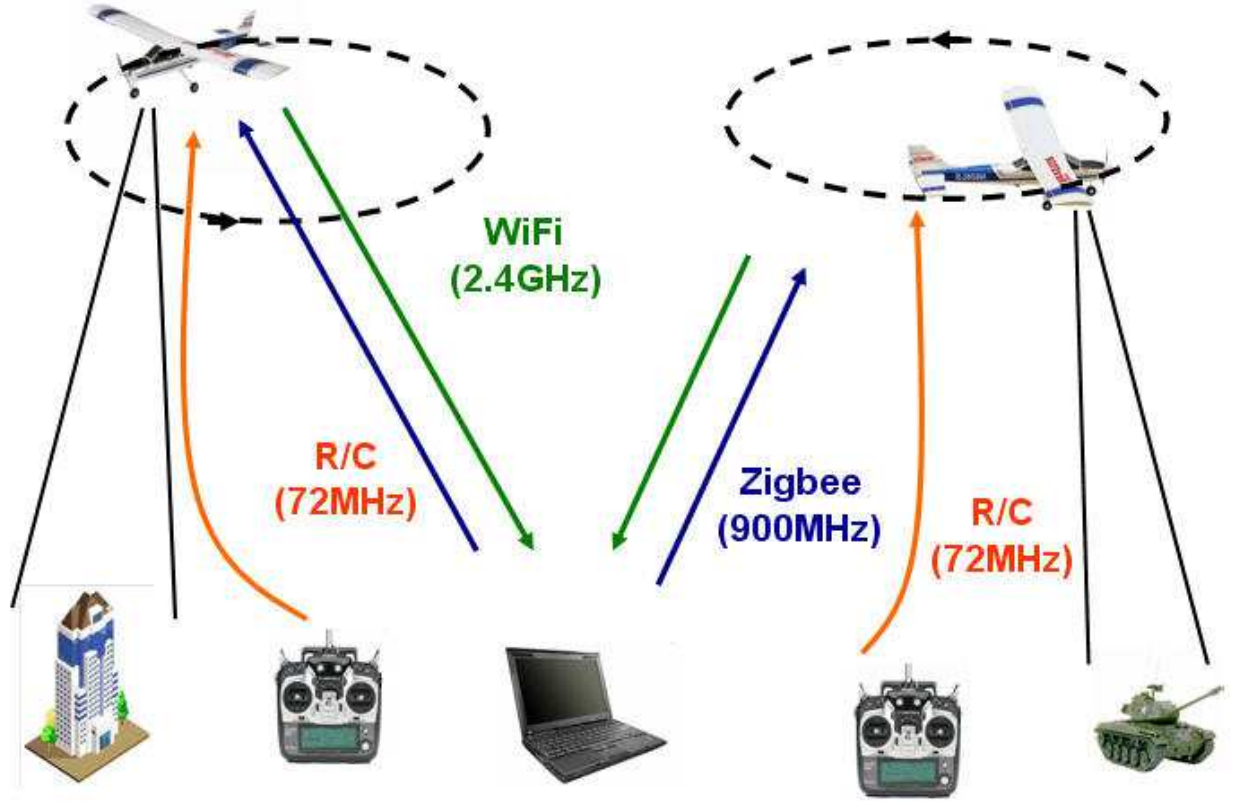


Fig. 1. Horus project diagram.

of the network is fixed throughout the experiment, but nodes are mobile and moving in fixed circular paths. This continuous movement results in a time-varying nature of the wireless channels and reveals the effectiveness of the implemented algorithms. We implement rate-distortion optimized algorithms and measure the system performance for these networks.

#### A. System Simulation

We have setup a simulation of Horus network in the OMNeT++ network simulator framework [13] using the MiXiM package [12]. To accurately mimic and evaluate RDO behavior in the Horus setup, we model both the time-varying nature of the network topology as well as the modified protocol stack including our RDO layer on top of the OMNeT++ component library. Each AAV node is described using a standard OMNeT++/MiXiM network stack consisting of a physical layer, a MAC-L layer and an application layer running the RDO optimized video streaming. Furthermore, we simulated mobile, time-varying network topologies using the circular motion module of OMNeT++.

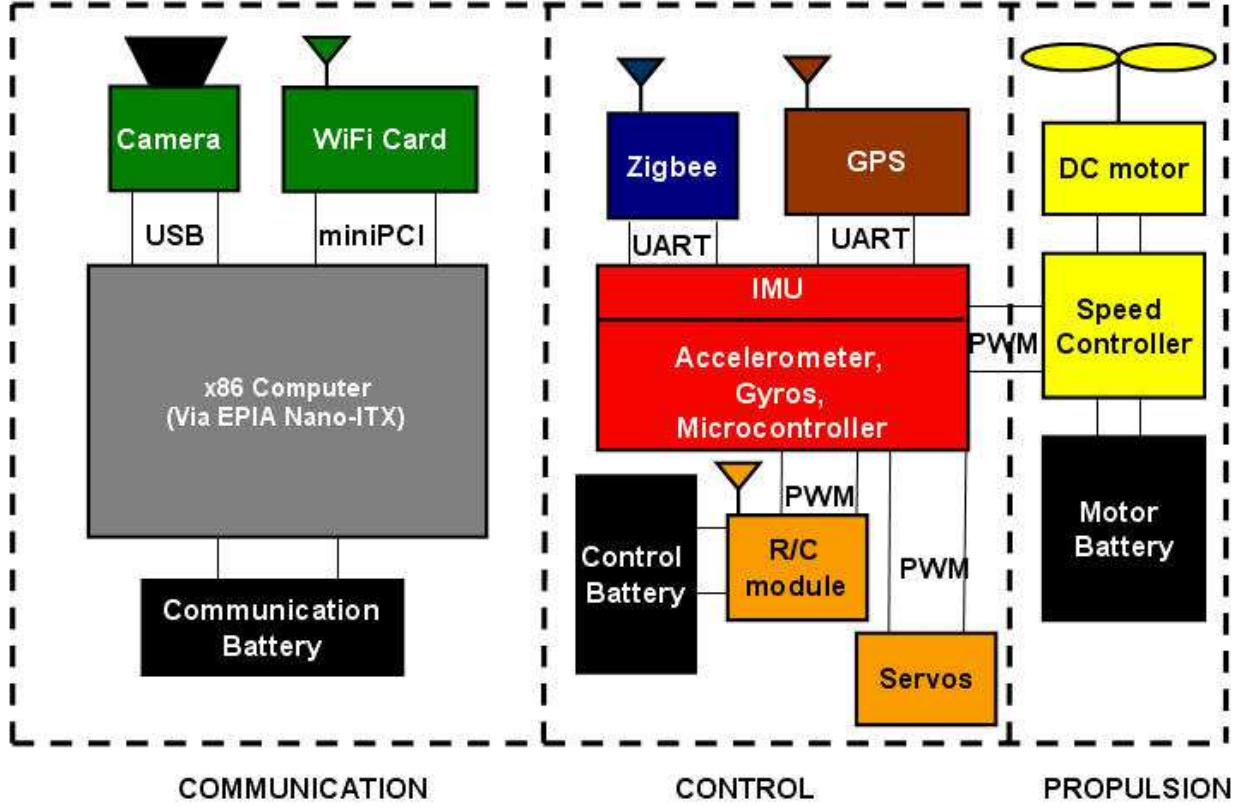


Fig. 2. AAV block diagram.

### B. System Implementation

In the following, we describe the actual implementation of our initial realization of the Horus network. We are considering AAVs as aerial nodes that form the network under test, see Figure 1. The AAVs are controlled manually during take-off and landing using a remote control (R/C) module running at 72MHz. In the air, the AAVs are operating under automatic control by the on-board inertial measurement unit (IMU). The IMU uses stored way-points and GPS<sup>4</sup> to follow a preprogrammed path. The path can be changed while the AAVs are in the air by uploading new waypoints from the ground laptop to the on-board IMU via a Zigbee radio link at 900MHz.

Due to the nature of AAV nodes, there are constraints on the weight and dimensions of the components, as well as the power consumption, which directly affects the possible transmission range. We divide the AAV node architecture into three subsystems for propulsion, control, and communication, see Figure 2<sup>5</sup>. The following lists represent the components that are used to construct our system nodes and their corresponding functions. The communication subsystem consists of the following components:

<sup>4</sup>GPS stands for Global Positioning System.

<sup>5</sup>UART, PWM, and USB stands for Universal Asynchronous Receiver/Transmitter, Pulse-Width Modulation, and Universal Serial Bus, respectively.

- 1) **Via EPIA Nano-ITX [17]:** is a x86 computer, which is the central unit for managing and operating the communication between nodes.
- 2) **Atheros WiFi radio [4]:** is a wireless card for the IEEE 802.11 2.4 GHz frequency band, which is used for wireless video transmission.
- 3) **Video Camera [10]:** captures videos for our measurements and could be used for location identification and object recognition in future extensions of Horus.
- 4) **Communication Battery:** provides an independent power source for the communication subsystem.

The control section consists of the following components:

- 1) **Zigbee [18]:** a radio for the IEEE 802.15 900 MHz frequency band used for receiving way-points for autonomous AAV navigation.
- 2) **IMU unit [2]:** controls the AAV movement during flight and sustains the required flight paths and mobile network topology.
- 3) **GPS unit [11]:** determines the location of the AAV for use by the autopilot in the IMU.
- 4) **R/C module [7]:** uses the 72 MHz frequency band and is the main manual ground control of the AAV. In Horus, it is used to control take-off and landing of the AAV.
- 5) **Servo motors [9]:** act on flaps and rudder for controlling the direction and orientation of the AAV.
- 6) **Control Battery:** independent power source for the control subsystem.

Finally, the propulsion section consists of the following components:

- 1) **DC motor [6]:** is the main moving force of AAV and is connected to the propeller.
- 2) **Speed Controller [1]:** controls the speed of the DC motor by regulating the input current.
- 3) **Motor Battery [16]:** main power source for AAV propulsion.

#### IV. LOW COMPLEXITY RDO WITH ACKs (LCRDO-ACK)

We realize the LCRDO-Ack algorithm as part of the application layer of our network. A conceptual block diagram for the LCRDO-Ack algorithm mapped to OSI network layers is shown in Figure 3. The video streams received from the camera are compressed into frames with different priorities. The *channel estimator* estimates the channel condition based on the received ACKs from previously sent packets. The *channel estimator* block is by default receiving ACKs from the destination in IEEE 802.11 networks. The *packet selector* block is responsible for selecting the suitable packet for transmission based on (1) the information received from the channel estimator and (2) the packet timestamp associated with each packet. In our case, the decision for retransmitting a packet or sending a new packet is made by the *packet selector* block according to the algorithm described in section IV-A.

##### A. LCRDO-Ack Algorithm

In MPEG4/MPEG2 video encoding, frames are compressed with different ratios and dependencies, giving each frame a different priority. Frames are divided into packets, where the amount of data and hence the number of

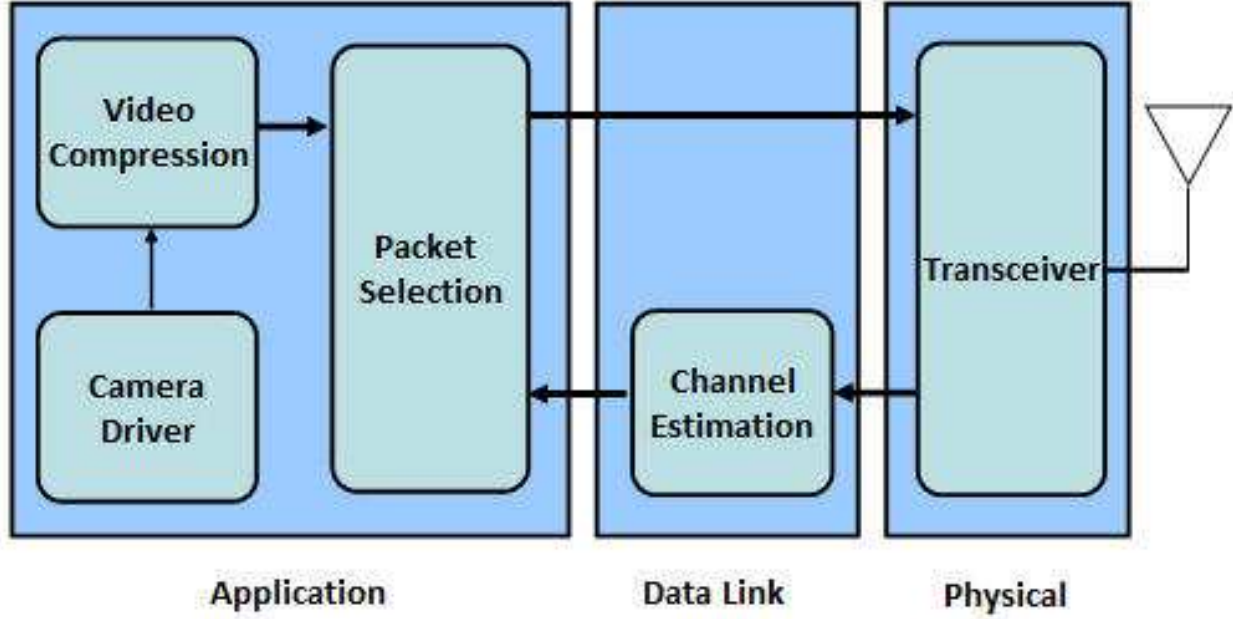


Fig. 3. Conceptual block diagram of the LCRDO-Ack algorithm.

packets typically increases with the frame priority. On the receiver side, these packets are recombined to form a frame that is then decoded. Therefore, losing a frame with high priority will lead to more deterioration in the quality of decoded video.

In line with existing video standards, we assume that the compressed video is composed of three types of frames: frames with first priority  $f_1$  (also known as *i frames*), frames with second priority  $f_2$  (*p frames*), and frames with third priority  $f_3$  (*b frames*). For compression purposes, the video frames are divided into Groups of Frames (GOF). Each GOF contains one  $f_1$  frame and a fixed number of  $f_2$  and  $f_3$  frames. The LCRDO-Ack algorithm that we use to optimize the video transmission is implemented in the application layer and is shown in Algorithm 1.

The algorithm consists of two main sections, *timestamp check* and *packet selection*. In the timestamp check, the algorithm starts by comparing the current time with the timestamp of the current GOF,  $t_{GOF\_timestamp}$ . If the GOF timestamp is not yet reached, packet selection is performed. Otherwise, the algorithm aborts the current GOF and advances to the next one. In packet selection (i.e. within the same GOF), the transmitter first sends  $f_1$  frame packets. Each  $f_1$  packet is (re-)transmitted until it is successfully sent and the algorithm can switch to the next one. After finishing the transmission of all  $f_1$  packets,  $f_2$  and  $f_3$  packets are sent. These lower priority packets are transmitted without waiting for feedback. This avoids wasting time in packet retransmissions, acknowledgements and reception times for these less important packets. At the end of packet selection, the packet ID is incremented to send the next packet until all packets in the current GOF are transmitted or a timeout is reached.

The LCRDO-Ack algorithm is implemented on top of standard protocol stacks. As an additional optimization we can, however, modify the 802.11 MAC-L to further improve overall real-time performance. Specifically, when



---

**Algorithm 1** LCRDO-Ack Algorithm

---

```
loop
  if  $t < t_{GOF\_timestamp}$  then
    if  $ID_{frame} = 1$  then
      if  $ID_{pkt} > ID_{finalf_1pkt}$  then
         $ID_{frame}++$  {switch to next frame}
      else
        transmit  $f_1$  packet
        if success then
           $ID_{pkt}++$  {switch to next  $f_1$  packet}
        end if
      end if
    else if  $ID_{pkt} \leq ID_{finalGOFpkt}$  then
      transmit  $f_2$  and  $f_3$  packets without retransmission
       $ID_{pkt}++$  {switch to next packet}
    end if
  else
     $t_{GOF\_timestamp} += \Delta t_{GOF}$  {start new GOF interval}
     $ID_{frame} \leftarrow 1$  { $ID_{frame} = 1$  is dedicated for i frame in GOF}
     $ID_{pkt} \leftarrow ID_{finalGOFpkt} + 1$  {packet ID set to first packet in GOF}
  end if
end loop
```

---

the MAC-L receives a packet with a  $f_1$  flag, it uses its default behavior to retransmit the packet up to 3 times until an ACK is received. If no ACK is received after 3 tries, the MAC-L reports a drop to the application layer. By contrast, in case of  $f_2$  or  $f_3$  packets, the RDO algorithm does not require feedback about transmission success and a modified MAC-L can transmit the packets only once without waiting for any ACK. This further reduces overall overhead and latencies.

### B. Distortion Measure

We investigate the LCRDO-Ack algorithm using a multiplicative distortion measure. The multiplicative distortion  $D_m$  is initially set to the maximum distortion level  $D_0$ . When frames are successfully received, the distortion decreases by the number of frames of type  $i$ ,  $N_{f_i}$ , multiplied by all the frames of higher priority successfully received within a GOF. This gives zero weight if frames of higher priority have not been successfully received.

$D_m$  is evaluated every GOF as

$$D_m = D_0 - N_{f_1} \left( 1 + N_{f_2} (1 + N_{f_3}) \right); \quad 0 \leq D_m \leq D_0.$$

We also define the sum of multiplicative distortion  $D_M$  as the sum of  $D_m$  for all transmitted GOFs:

$$D_M = \sum_{GOF} D_m.$$

$D_M$  is the distortion measure we use to compare different transmission protocols.

### C. Experimental Setup

For our experiments, we assume a topology in which two or three AAVs (hosts) move in fixed circular patterns with a radius of 40 meters and a distance of 380 meters between the circles centers, see Figure 1. Nodes send data packets in a one-hop fashion over a 802.11 wireless connection, where the source node transmits packets directly to the destination node. Next to the transmission under test, we include a third node that simultaneously transmits other packets not related to the main video stream. This setting allows us to analyze RDO transmission in the presences of high interference and consequently when experiencing a large packet loss.

We simulated this setup in OMNET++ using a simple path loss channel model as the one most closely resembling AAV-to-AAV conditions with little to no fading and no shadowing effects. The continuous motion of the nodes in circular paths leads to time-varying channel effects and a network packet drop rate that depends on the relative position of the AAVs. We use different path loss exponents  $\alpha$  to model and experiment with normal and worst case channel conditions. For worst-case analysis, we assume an exponent  $\alpha$  of 3.7. Together with interference from a third node as described above, we observe an overall packet drop rate of 40%, which allows for comparison of various transmitters under realistic conditions.

For testing the LCRDO-Ack algorithm, we compare it against conventional transmission algorithms. Overall, we define three types of transmitters:

- 1) *Transmitter without ACK*: transmits every packet without waiting for an ACK from the receiver.
- 2) *Transmitter with ACK*: retransmits every packet until it receives an ACK for each packet.
- 3) *Transmitter running the LCRDO-Ack algorithm*: implements the LCRDO-Ack algorithm described previously.

### D. Results

To compare different transmitters, we run the network simulator for each transmitter under the exact network conditions mentioned in the previous subsection. We specify 3500 packets to be transmitted from the source to the destination node. For simplicity, we fix the number of packets per frame for a given priority frame. For the payload parameters, the number of priority  $f_1$ ,  $f_2$ , and  $f_3$  frames per GOF are 1, 2, and 6, respectively, and the number of packets per frame  $f_1$ ,  $f_2$ , and  $f_3$  are 50, 20, and 10, respectively.

Our performance investigation for this problem includes both a network measure (e.g. number of packets drops) and an optimization measure (e.g. quality of the received media). We illustrate both using an easy-to-visualize proxy

variable, which is a counter at the receiver. This counter counts the number of successfully received packets and is incremented until the end of the frame is reached, upon which the counter is reset to zero. Counting then starts in the same manner for the next frame, and so on. The counter can determine if the received packet is in the current frame or not by checking the packet ID number associated with it. This way, plotting the counter values over time visualizes the performance of the receiver with respect to the video frames. These values are later used to measure the distortion.

The No-ACK Transmitter sends packets continuously without receiving any ACKs from the receiver. This leads to loss of packets with equal probability for different priority frames, which results in a 40% loss of the first priority packets essential to decode the other packets sent within a GOF. Due to this behavior, about 40% of frames are not completely received leading to high distortion, for more details see Figures in [19].

The ACK Transmitter sends new packets only after receiving an ACK for the previous packet, and it otherwise continues to retransmit the same packet. Therefore, all the frame packets within the current GOF timestamp are received successfully regardless of their priority. The frames received after their GOF timestamp are dropped, but at the same time cause more delay to build up with time. This accumulative delay is caused by retransmission and ACKing of packets of lower priority. Due to this delay, the number of frames that are lost increases as the transmission continues, causing a large degradation in the video quality over time. This leads to a significant increase in the distortion at the end of simulation. The number of frames lost per GOF increase as the transmission continues, leading to high distortion  $D_M$ , see Figures in [19]. This transmitter experiences the highest distortion when the simulation is allowed to run for a sufficiently long time.

The LCRDO-Ack transmitter is designed to minimize the distortion measure and give better performance than conventional transmitters. It retransmits until an ACK is received only for the first priority packets. Second and third priority packets are sent without waiting for an ACK from the destination. This guarantees that frames of priority  $f_1$  will be received at the receiver even under bad channel conditions, which is the case in our simulation. The second and third priority frames,  $f_2$  and  $f_3$  respectively, are dropped with about 40% probability. Overall, the LCRDO-Ack transmitter guarantees a minimum video quality at the receiver side and gives a better distortion than other transmitters.

In all simulated cases, the average packet drop rate is 40%. In the LCRDO-Ack case, it is guaranteed that most of these drops are lower priority packets, which affects transmission quality less and makes the LCRDO-Ack algorithm more robust. In the No-ACK and ACK transmitters, these dropped packets can be of any type of packet priority. Therefore, in these two cases, the 40% drop rate significantly affects transmission quality.

## V. LOW COMPLEXITY RDO WITH BEACONING AND ADAPTIVITY (LCRDO-BEACON AND LCRDO-ADAPTIVE)

We implemented LCRDO-Beacon and LCRDO-Adaptive algorithms in our physical testbed setup as described in section III-B. On the Via EPIA computer mounted inside the AAV, we run a Ubuntu 10.10 Linux operating systems with kernel version 2.6.35. We use the GStreamer open source multimedia framework [5] for video compression

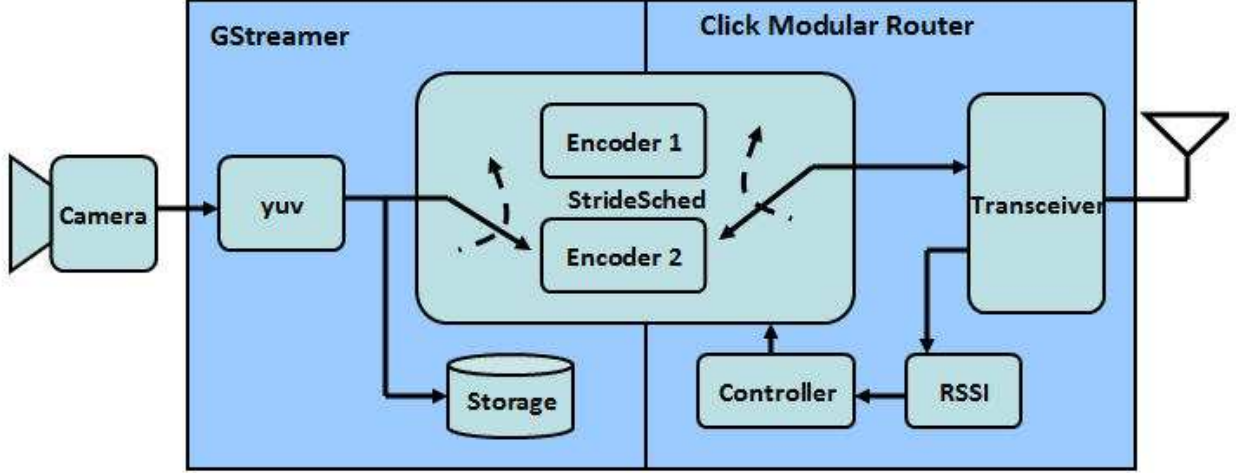


Fig. 4. Universal block diagram of the implemented LCRDO algorithm at the transmitter.

and the Click Modular Router open source network stack [15] for implementing our transmission algorithm. The operating system used for the ground station (i.e. the ground laptop) is Ubuntu 10.10 with Linux kernel 2.6.35. We implement the LCRDO-Beacon algorithm in connection with MPEG2 video compression and the LCRDO-Adaptive algorithm for both MPEG2 and MJPEG compressions.

#### A. LCRDO-Beacon Algorithm

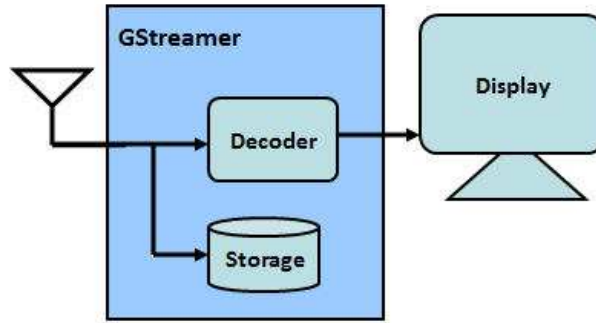


Fig. 5. Universal block diagram of the implemented LCRDO algorithm at the receiver.

The LCRDO-Beacon algorithm, used in the system implementation, is computationally less complex than the LCRDO-Ack algorithm (Algorithm 1) used in the system simulation. The reason for further reducing the complexity of the algorithm are the high computational demands of MPEG2 video encoding, which requires most of the computation power of Via EPIA computer. The modifications of switching from ACKs to beacons for channel state measurement ensure a more reliable real-time performance and concurrently show significant improvement in the

---

**Algorithm 2** Implemented LCRDO Algorithm

---

```
loop
    transmit packets received from the selected/active encoder
    if transceiver receives a beacon then
        if  $RSSI < X_1$  then
            Clear Queue1 and Queue2
            Switch to Encoder 2
        else
            if  $RSSI > X_2$  then
                Clear Queue1 and Queue2
                Switch to Encoder 1
            end if
        end if
    end if
end loop
```

---

general performance with respect to unoptimized methods, as will be shown in section V-F.

The universal block diagram of the implemented LCRDO algorithm at the transmitter is show in Figure 4. For LCRDO-Beacon algorithm at the transmitter, the camera driver outputs raw video, which is resized to 160x120 (without loss of generality, this resolution is chosen to minimize the computation on the Via EPIA computer; the algorithm applies equally for higher resolutions). A stride scheduler chooses between different parameters for encoding the raw video into an MPEG2 stream. We realize two different encoders:

- 1) MPEG2 encoder with GOF<sup>6</sup>=5 and an overall frame rate of 25 frames per second (f/s). This corresponds to sending i, p and b frames with an i frame rate of 5 f/s.
- 2) MPEG2 encoder with GOF=1 and an overall frame rate of 5 f/s. This corresponds to sending i frames only at a rate of 5 f/s.

Note that overall, this setup is equivalent to a general RDO architecture as described in section IV (Figure 3), where packets of p and b frames of a single, fixed MPEG2 encoder (running at 25 f/s) are selectively dropped depending on the chosen transmission policy. An equivalent implementation that alternates between two separate encoders as controlled by a stride scheduler was chosen due to limitations of the GStreamer-internal architecture.

The two outputs of the two MPEG2 compression and packet selection blocks are fed into the transmitter via two queues, *Queue1* and *Queue2*. These queues are not drawn to simplify the block diagram. For later comparison purposes, a high quality reference copy of the original video is stored in compressed MPEG2 form (with GOF=5

<sup>6</sup>Group Of Frames (GOF) is the group of video frames that starts by an i frame then proceeded by p and b frames only. The MPEG video file is a sequence of GOFs; e.g. GOF=5 has one i frame and four p and b frames, while GOF=1 has only i frames and no p and b frames.

and 25 f/s) using GStreamer. Inside the Click Modular Router, a transceiver realizes wireless transmission of encoded videos and wireless reception of signal strength beacons. The beacons received in the transceiver are passed through a Received Signal Strength Indicator (RSSI<sup>7</sup>) block that decodes the RSSI value and passes it on to a controller block. Finally, the controller block determines packet selection and controls the stride scheduler by executing Algorithm ??.

The implemented LCRDO algorithm is shown in Algorithm 2. In the LCRDO-Beacon algorithm case, the controller switches to the Encoder 2 (i frames only) in the video compression block whenever the RSSI drops below a value  $X_1$ . Likewise, if the RSSI rises above a value  $X_2$ , the controller switches to select the Encoder 1 (i, p and b frames) in the video compression block. The graph of the RSSI on the x-axis and the active encoder on the y-axis form a sharp hysteresis loop (figure removed due to figures limit). In the switching instant, all the contents of *Queue1* and *Queue2* are cleared. This is done to avoid sending any residual packets in the queue when switching back and forth between the different encoding modes, ensuring reliable real-time performance.

The universal block diagram of the implemented LCRDO algorithm at the receiver is show in Figure 5. For LCRDO-Beacon algorithm at the receiver, the received signal is decoded using a standard MPEG2 decoder and displayed on the monitor of the ground station (laptop). At the same time, the received video is stored in compressed MPEG2 format for later analysis.

#### B. LCRDO-Adaptive Algorithm for MPEG2

In addition to selectively dropping packets, a generalized method for performing RDO and improving the received video quality is to co-design RDO-type packet selection with adaptive video encoding. In such an approach, the video encoding rate is adjusted to the transmission rate in a distortion-optimized way. In addition to improving video quality, adapting encoding parameters to rate variations can significantly reduce average computational requirements in the real-time video encoder. Similar to the LCRDO-Beacon algorithm, such an adaptive approach has an operating mechanism in which the video encoder switches between different modes. Both algorithms require predetermined thresholds, i.e.  $X_1$  and  $X_2$ , used in a sharp hysteresis loop. In the LCRDO-Beacon algorithm, the channel state measurement determines when to transmit both independent and dependent or when to drop dependent and only transmit independent frames. By contrast, for the adaptive algorithm, the channel state measurements determine when to transmit video at high quality, i.e. with high bit rate, and when to transmit video at low quality, i.e. with lower bit rate. Both algorithms require channel state measurements and seek to minimize distortion and maximize video quality.

The block diagram of the low complexity RDO with adaptive co-design (LCRDO-Adaptive) algorithm for MPEG2 transmissions is similar to the block diagram of LCRDO-Beacon with the exception that the stride scheduler allows choosing between two different MPEG2 encoders for video compression:

<sup>7</sup>The Atheros based card returns an RSSI value of 0 to 127 (0x7f) with 128 (0x80) indicating an invalid value. There is no specified relationship of any particular physical parameter to the RSSI reading.

- 1) MPEG2 encoder with 5 frames per GOF, a frame rate of 25 f/s, and unlimited bit rate.
- 2) MPEG2 encoder with 5 frames per GOF, 25 f/s frame rate, and 100 kbps transmission rate.

The block diagram of the receiver is similar to the LCRDO-Beacon receiver.

### C. LCRDO-Adaptive Algorithm for MJPEG/SMOKE

The attractive property of Motion JPEG (MJPEG) video compression is the very low computational complexity compared to MPEG2 video compression. This comes at the expense of lower compression leading to higher bandwidth requirements. Furthermore, MJPEG compression is characteristic by all frames being independent, whereas MPEG2 compression has both independent and dependent frames. A variant of MJPEG compression is the SMOKE codec [14], which includes both JPEG frames and delta frames. JPEG frames are key-frame that are each followed by  $N - 1$  delta frames. JPEG frames are independent while delta frames are constructed according to a motion estimation threshold using the key-frame. This threshold specifies how much each 16x16 block of pixels may differ before a new block is generated. A large value of the threshold causes more blocks to stay the same for more frames, decreasing bandwidth usage, but producing less accurate output. Likewise, a small number of delta frames between key-frames increase received video quality at the cost of a higher bit rate, and vice versa.

The block diagram of the LCRDO-Adaptive for MJPEG/SMOKE transmitter is similar to the block diagram of LCRDO-Beacon with the following exceptions. The camera driver outputs raw video which is resized to 320x240 with frame rate of 10 f/s. A stride scheduler chooses between different parameters for encoding the raw video into an MJPEG/SMOKE stream. We realize two different encoders:

- 1) MJPEG/SMOKE encoder with high JPEG image quality of 80% and  $N=8$ .
- 2) MJPEG/SMOKE encoder with low JPEG image quality of 30% and  $N=8$ .

The storage block stores high quality MJPEG/SMOKE compressed video (with JPEG image quality of 80%,  $N=8$  and 10 f/s) as reference for later analysis.

On the receiver side, see Figure 5, the received signal is decoded using MJPEG/SMOKE decoder and displayed on the monitor of the ground station (laptop). At the same time, the received compressed video is stored as MJPEG/SMOKE format for later analysis.

### D. Distortion Measures

The state of the art metric for comparative assessment of video quality is the MOTion-based Video Integrity Evaluation (MOVIE) Index [30]. However, since we experience frame drops in our video transmissions, it is difficult for us to apply the MOVIE index directly. Frame drops result in different video duration between the reference video stored in the transmitter and the video to be evaluated in the receiver. Therefore, we measure distortion in our experiments differently. First, we measure the number of frame drops experienced in the transmission, which represents the temporal loss in the received information. Second, we measure the spatial loss in the video by slicing the video into images/frames and comparing manually selected, representative best and worst received images/frames to their corresponding source images/frames.

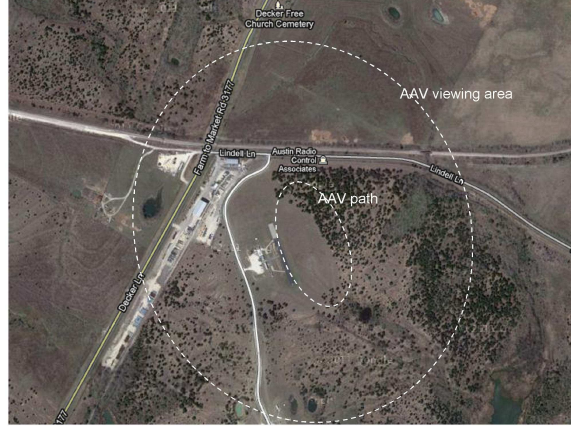


Fig. 6. Map of the actual site used for running Horus experiments. The internal circle shows the AAV flight path. The outer circle shows the viewing area of the camera attached to the AAV.

We qualify temporal distortion by comparing the difference in duration between the video viewed at the receiver and the high quality reference video at the transmitter. The high quality original video file is compressed by the encoder at the source while simultaneously being transmitted to the destination. At the destination, a copy of the video received is stored in compressed form while also being viewed in real-time.

In addition, the MPEG2 decoder performs inter-frame estimation for the missing packets in the frame. The estimation process causes some of the frames to be distorted. To capture such frame distortions, we introduce an additional spatial metric. The spatial metric is intended to provide an approximate measure of the distortion in received frames when applied to the best and worst manually selected frames, as will be described in section V-F. We measure the spatial distortion by applying the Structural SIMilarity (SSIM) [33] index to the best and worst frames successfully received/reconstructed at the receiver. The SSIM index is a method for measuring the humanly perceived similarity between two images. It is a reference-based metric, where the assessment of image quality is based on a distortion-free reference image. We choose SSIM because it outperforms traditional methods, such as peak signal-to-noise ratio (PSNR) and mean squared error (MSE), which have proven to be inconsistent with human perception.

#### E. Experimental Setup

In Horus, we consider two main network topologies:

- 1) Unicast network topology: one AAV moving in a circular pattern. The AAV is the source that records video of the landscape and transmits this video to the destination. The destination is a laptop in the ground station running Linux.
- 2) Multiple unicast network topology: two AAV moving in two separate circular paths. The AAVs are both sources that record video of the landscape and transmit them to one destination (as shown in Figure 1). The destination is a laptop in the ground station running Linux.



Note that a multicast network topology is easier to implement than a multiple unicast network topology. Two sources transmitting to one destination requires twice the bandwidth of a network in which one source transmitting to two destinations. Therefore, if a multiple unicast network topology is implementable and reliable using our proposed algorithms, then multicast network topology will almost certainly be implementable and reliable.

We run the flight experiment at Lester Field [3] in Austin, Texas. In the flight experiments, the AAV records video of the landscape and transmits this video in real-time to the ground station (laptop). Throughout each flight, AAVs pass through four different flight phases, which correspond to different characteristics of the transmitted video:

- A) The AAV is stationary on the ground and near the laptop (the ground station). This phase takes place during AAV initialization and preparation.
- B) The AAV is moving slowly and within close range of the laptop. This phase takes place when moving the AAV to the take-off runway.
- C) The AAV is moving relatively fast and with moderate range of the laptop. This phase takes place at take-off and landing. In this phase, the recorded camera video is changing rapidly.
- D) The AAV is moving fast and far away from the laptop. This phase takes place when the AAV is in the air.

The AAV passes through these phases every time we run a flight experiment to test different transmission algorithms. In total, we conducted three different flight experiments:

- 1) Evaluating the performance of the LCRDO-Beacon algorithm with respect to unmodified MPEG2 video transmissions. We test the following:
  - a) *Unmodified transmitter with only i frames*: MPEG2 codec with GOF=1 and 5 f/s.
  - b) *Unmodified transmitter with all i, p and b frames*: MPEG2 codec with GOF=5 and 25 f/s.
  - c) *Transmitter running the LCRDO-Beacon algorithm*: alternating MPEG2 codecs as discussed in section V-A.
- 2) Evaluating the performance of the LCRDO-Adaptive algorithm for MPEG2 with respect to unmodified MPEG2 video transmissions. We test the following:
  - a) *Unmodified transmitter with 100 kbps*: MPEG2 codec with 100 kbps transmission rate and GOF=5 and 25 f/s.
  - b) *Unmodified transmitter with unlimited rate*: MPEG2 codec with unlimited rate and GOF=5 and 25 f/s.
  - c) *Transmitter running the LCRDO-Adaptive algorithm for MPEG2*: adaptive MPEG2 codec as discussed in section V-B.
- 3) Evaluating the performance of the LCRDO-Adaptive algorithm for MJPEG/SMOKE with respect to unmodified MJPEG/SMOKE video transmissions. We test the following:
  - a) *Unmodified transmitter with low quality*: MJPEG /SMOKE codec with low quality of 30% and  $N=8$ .
  - b) *Unmodified transmitter with high quality*: MJPEG /SMOKE codec with high quality of 80% and  $N=8$ .
  - c) *Transmitter running the LCRDO-Adaptive algorithm for MJPEG*: adaptive MJPEG/SMOKE codec as discussed in section V-C.

*Remark 5.1:* The hysteresis threshold values that we use in all the flight experiments are  $X_1=30$  and  $X_2=50$ .

TABLE I  
RECEIVED VIDEO DURATION AND AVERAGE SSIM INDICES

Experiment	Video duration	Average SSIM
1(a) Send only i frames	0.68	0.773
1(b) Send all i, p, and b frames	0.53	0.770
1(c) LCRDO-Beacon	0.76	0.777
2(a) Send at 100kbps	0.71	0.717
2(b) Send at unlimited rate	0.58	0.789
2(c) LCRDO-Adaptive for MPEG2	0.76	0.762
3(a) Send at low quality	0.43	0.939
3(b) Send at high quality	0.27	1
3(c) LCRDO-Adaptive for MJPEG	0.49	0.959

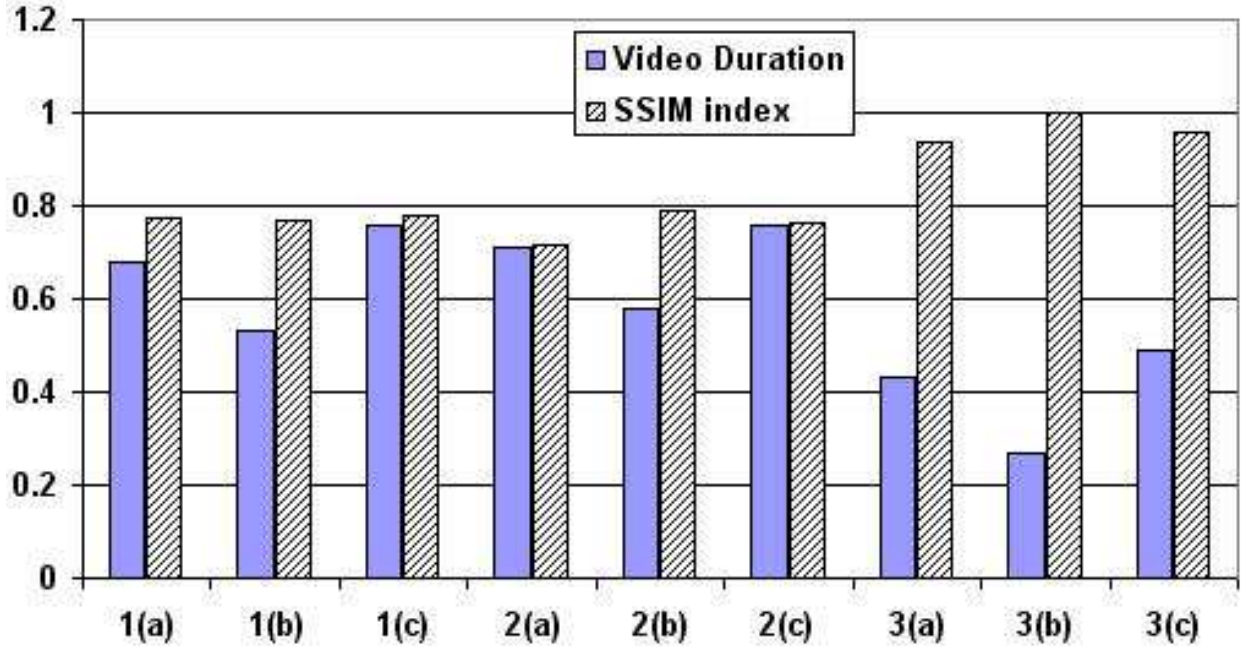


Fig. 7. Received video duration and average SSIM indices.

#### F. Results

For our experiments, we analyze both temporal and spatial distortion. For temporal distortion, we compare the video at the receiver/destination and the video at the transmitter/source with respect to video duration. For spatial distortion, we compare the SSIM index of 10 highest and 10 lowest spatially distorted video frames received at the destination. The frame selection is done manually by eye inspection. The number of frames selected from the flight phases A, B, C, and D are 2, 4, 4, and 10, respectively. The SSIM index is calculated for each frame with respect to the corresponding source frame. Then, the average, i.e. arithmetic mean, of SSIM indices for all the selected 20 frames is calculated for each case in each flight experiment.

Table I and Figure 7 show the temporal distortion results for our experiments. For every experiment, the fraction of the received video duration is highest for case (c). Hence, we can conclude that a better optical flow of received video is observed when applying the LCRDO algorithms compared to the other two cases for each experiment. Therefore, the temporal distortion observed by the user is minimal in the LCRDO algorithms. This corresponds to our experiences during real-time field observation.

*Remark 5.2:* When we run the same case in the same experiment under the same conditions multiple of times, we observe a 5% difference in the results. Therefore, we can conclude that all the results presented here have an accuracy around 95%.

In terms of spatial distortions, we observed that the average SSIM indices for case (a), (b), and (c) within each experiment are approximately equal, see Table I and Figure 7. We conclude that the average SSIM index is more dependent on the compression type compared to the transmission algorithm. Nevertheless, spatial distortions in the RDO case are always less than when streaming with a low quality encoder, approaching the level of high quality encoding yet with much better temporal behavior. The MJPEG/SMOKE codec outperforms MPEG2 codec in terms of its average SSIM index. This is due to inter-frame estimation performed in the MPEG2 decoder, which causes partially received frames to be viewed as distorted frames. By contrast, partially received frames are dropped in the MJPEG/SMOKE decoder, leading to higher temporal distortions instead.

Finally, we conducted a flight experiment for the multiple unicast network topology mentioned in section V-E. The two AAVs transmit two different video signals simultaneously to a common ground station (laptop) using the LCRDO-Adaptive for MJPEG algorithm. The main observation is that we can display both videos with acceptable optical flow. Successful reception of multiple unicast flows should open more possibilities for future work in building larger AAV networks that can transmit/receive videos to/from multiple destinations/sources. As discussed previously, any implementation of the more complex multiple unicast case should easily transfer into a multicast environment.

## VI. CONCLUSION

Rate distortion optimization has its origins in information theory [26]. Rate distortion represents the minimum rate required to compress information at a distortion level of  $D$ . The rate distortion function with and without state is well known [26], and can be computed for most cases algorithmically and in some cases in closed form. Our rate distortion algorithm thus aims at bridging theory and practice, by building an algorithmic framework for real-time rate distortion optimization that is low complexity and thus practically viable over an AAV testbed.

This paper constructs a reliable wireless networks test-bed for testing new wireless networks protocols called Horus. In this test-bed, we implement the problem of streaming packetized media over a wireless network using a rate-distortion optimized algorithm. We compare different transmission algorithms both in simulation and implementation. We give a comparative study to the design trade-offs to be considered to achieve a reliable and optimized video transmission. The main intuition that emerges from Horus is that in order to provide a good real-time video transmission performance, one should consider both the computation power and the bandwidth limitations. For low computation power, we find that MJPEG/SMOKE as the best choice. For moderate computation power, we have

MPEG2 as the more suitable solution. For limited bandwidth but high computation power, MPEG4-H264 could potentially be the more suitable solution because it has better utilization of the available bandwidth compared to MJPEG/SMOKE and MPEG2. This is the trade-off between bandwidth and computation limitation and the video quality that can be achieved.

## VII. FUTURE WORK

In this paper, we considered optimized video transmission. The current work on this topic is looking at extending this work to long term evolution (LTE) systems, say, optimal resource allocation in a cellular system. We are looking at the viability of the rate distortion optimization problem in joint optimization with resource allocation of video applications in cellular system, along the lines of [21]–[23], [31], [32], to improve the quality of experience of mobile users. In addition, we plan on the inclusion of wireless communication capacity [27], and interference alignment methods for MIMO communication systems [20].

## REFERENCES

- [1] 80A Phoenix ICE HV speed controller with Data Logging Capability, <http://www.truerc.com>.
- [2] ArduPilot Mega, <http://diydrones.com/profiles/blogs/ardupilot-mega-home-page>.
- [3] Austin RC, <http://www.austinrc.org/index.html>.
- [4] CM9-GP: 802.11 a/b/g 108Mbps wifi mini-PCI module, <http://www.unex.com.tw/product/cm9-gp>.
- [5] GStreamer: open source multimedia framework, <http://gstreamer.freedesktop.org/>.
- [6] Hacker Brushless A40-10L, <http://www.aero-model.com>.
- [7] Hitec Eclipse 7 7-Channel FM Transmitter/Spectra, <http://www.scribd.com>.
- [8] Horus: A Wireless Network of AAVs, <http://theseus.ece.utexas.edu/horus/>.
- [9] HS-322 Deluxe Standard Servo, <http://www.hobby-lobby.com/hs-322-standard-servo-2786-prd1.htm>.
- [10] Logitech QuickCam Pro 4000, <http://www.amazon.com/Logitech-961239-0403-QuickCam-Pro-4000>.
- [11] MediaTek MT3329 GPS 10Hz, <http://store.diydrones.com/MediaTek-MT3329-GPS-10Hz-p/mt3329-01.htm>.
- [12] MiXiM OMNeT++ Framework Project, <http://mixim.sourceforge.net/>.
- [13] OMNeT++ Network Simulation Framework, <http://www.omnetpp.org>.
- [14] Smoke Encoder, <http://www.flumotion.net/doc/flumotion-manual/en/0.6.0/html/>.
- [15] The Click Modular Router Project, <http://read.cs.ucla.edu/click/click>.
- [16] Thunder Power 3S4P 8000 mAh Li-Poly Packs TP8000-3S4P, <http://aeromicro.com>.
- [17] Via EPIA Nano-ITX, <http://www.via.com.tw>.
- [18] XBee-PRO DigiMesh 900 Mesh RF Modules, <http://www.digi.com/products/wireless/zigbee-mesh/xbee-digimesh-900>.
- [19] A. Abdel-Hadi, J. Micheal, A. Gerstlauer, and S. Vishwanath. Real-time optimization of video transmission in a network of aavs. *IEEE 74th Vehicular Technology Conference, San Francisco, United States*, 2011.
- [20] A. Abdel-Hadi and S. Vishwanath. On multicast interference alignment in multihop networks. *IEEE Information Theory Workshop, Cairo, EGYPT*, 2010.
- [21] Ahmed Abdel-Hadi and Charles Clancy. A Robust Optimal Rate Allocation Algorithm and Pricing Policy for Hybrid Traffic in 4G-LTE. In *PIMRC*, 2013.
- [22] Ahmed Abdel-Hadi and Charles Clancy. A Utility Proportional Fairness Approach for Resource Allocation in 4G-LTE. In *ICNC Workshop CNC*, 2014.
- [23] Ahmed Abdel-Hadi, Charles Clancy, and Joseph Mitola. A Resource Allocation Algorithm for Multi-Application Users in 4G-LTE. In *MobiCom Workshop*, 2013.

- [24] D. Agrawal, T. Bheemarjuna Reddy, C. Siva, and Ram Murthy. Robust Demand-Driven Video Multicast over Ad hoc Wireless Networks. In *Proc. of BroadNets '06*, October 2006.
- [25] P. A. Chou and Z. Miao. Rate-Distortion Optimized Streaming of Packetized Media. *IEEE Transaction on Multimedia*, May 2005.
- [26] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.
- [27] J. Jose, A. Abdel-Hadi, P. Gupta, and S. Vishwanath. Impact of mobility on multicast capacity of wireless networks. *IEEE InfoCom, San Diego, CA, USA*, 2010.
- [28] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li shiuan Peh, and Daniel Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proc. of ACM (ASPLOS-X)*, pages 96–107, October 2002.
- [29] Ting Liu, Christopher M. Sadler, Pei Zhang, and Margaret Martonosi. Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet. In *Proc. of MobiSYS '04*, pages 256–269. ACM Press, 2004.
- [30] K. Seshadrinathan and A. Bovik. Motion tuned spatio-temporal quality assessment of natural videos. *IEEE Transactions on Image Processing*, pages 335–350, Feb. 2010.
- [31] Haya Shajiah, Ahmed Abdel-Hadi, and Charles Clancy. Utility Proportional Fairness Resource Allocation with Carrier Aggregation in 4G-LTE. In *MILCOM*, 2013.
- [32] Haya Shajiah, Ahmed Abdel-Hadi, and Charles Clancy. Spectrum Sharing between Public Safety and Commercial Users in 4G-LTE. In *ICNC*, 2014.
- [33] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, pages 600–612, Apr. 2004.
- [34] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware Design Experiences in ZebraNet. In *Proc. of SenSys '04*, November 2004.